Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1-6. (Cancelled)

7. (Currently Amended) A computer implemented method for exporting from graphical representations of business processes to structural text-based representations and for importing from structural text-based representations of business processes to graphical representations, the method comprising:

implementing [[the]] storage and maintenance of a set of features identifiable in graphical business process representations, each feature in the set of features having an associated pattern mapping defined relative to structural text-based representations,

identifying portions of an initial graphical representation as matching features in the set of features,

generating structural text-based representations of the identified portions of the initial graphical representation by applying the pattern mappings associated with the matching features to the identified portions of the graph-based representation,

identifying portions of an initial structural text-based representation of a business process as corresponding to pattern mappings associated with features in the set of features, and

generating graphical representations of the identified portions of the initial structural text-based representation by reference to the features associated with the pattern mappings corresponding to the identified portions of the initial structural ~~textbased~~ text-based representation, wherein the generated graphical representations are visually presented to a user.

8. (Currently Amended) The method of claim 7, in which the set of identifiable features and the associated pattern mappings, comprises at least five feature and pattern mapping pairs selected from the following set of pairs:

i. feature: synchronous/asynchronous processes; pattern mapping: a synchronous process representation comprises a <receive> activity as its input interface, and a <reply> activity as its output interface; an asynchronous process representation comprises a <receive> activity as its input interface, and an <invoke> activity as its output interface;

ii. feature: request/response activity; pattern mapping: an <invoke> activity with attributes inputContainer and outputContainer to specify input and output containers assigned to the activity;

iii. feature: one-way activity; pattern mapping: an <invoke> activity, with attribute inputContainer and no outputContainer;

iv. feature: empty node; pattern mapping: an <empty> activity defined by a naming convention including node name;

v. feature: block; pattern mapping: a <scope> activity with two <empty> activities nested within the <scope> activity to represent the input and output nodes in the block;

vi. feature: iteration; pattern mapping: a <while> activity having an attribute condition equivalent to the loop condition in the loop node of the iteration; two <empty> activities nested within the <while> activity to represent input and output nodes in the loop body of the iteration;

vii. feature: receive event; pattern mapping: a <pick> activity containing <onMessage> structures to define events accepted by the <pick> activity, corresponding to events defined in the receive event;

viii. feature: compensation; pattern mapping: a <compensationHandler> structure comprising an activity within the structure to compensate an execution failure;

ix. feature: correlation; pattern mapping: a <correlation> element having a correlation ID defined and referenced by a <correlationSet> element; the <correlation> element being nested within a <receive> activity representing an input node, and within all <pick> activities corresponding to one or more receive event nodes;

x. feature: variables; pattern mapping: containers;

xi. feature: fault handling; pattern mapping: a <catch> structure containing elements in a fault path if the fault is only thrown once; where the fault is capable of being repeatedly caught and thrown then

(a) if thrown internally: a <throw> activity; or

(b) if thrown externally: a <reply> activity; and

xii. feature: transition condition; pattern mapping: an attribute in a <source> element of a <link> element representing the transition[[;]] .

9. (Currently Amended) The method of claim 7, in which the ~~export generation code~~ generating step further comprises ~~conversion code for~~ converting Java code referenced in the initial graphical representation to XPath code in the generated structural text-based representation.

10-20. (Cancelled)

21. (Currently Amended) A computer implemented method for importing from structural text-based representations of business processes to graphical representations, the method comprising:

implementing the storage and maintenance of a set of features identifiable in graphical business process representations, each feature in the set of features having an associated pattern mapping defined relative to structural text-based representations,

identifying portions of an initial structural text-based representation of a business process as corresponding to pattern mappings associated with features in the set of features, and

generating graphical representations of the identified portions of the initial structural text-based representation by reference to the features associated with the pattern mappings corresponding to the identified portions of the initial structural ~~textbased~~ text-based representation, wherein at least one of the features is fault handing and the associated pattern mapping for the fault handling comprises a catch structure specifying fault name and container, and wherein the generated graphical representations are visually presented to a user.

22. (Currently Amended) The method of claim 21, in which the set of identifiable features and the associated pattern mappings, comprises at least five feature and pattern mapping pairs selected from the following set of pairs:

i. feature: synchronous/asynchronous processes; pattern mapping: a synchronous process representation comprises a <receive> activity as its input interface, and a <reply> activity as its output interface; an asynchronous process representation comprises a <receive> activity as its input interface, and an <invoke> activity as its output interface;

ii. feature: request/response activity; pattern mapping: an <invoke> activity with attributes inputContainer and outputContainer to specify input and output containers assigned to the activity;.

iii. feature: one-way activity; pattern mapping: an <invoke> activity, with attribute inputContainer and no outputContainer;

iv. feature: empty node; pattern mapping: an <empty> activity defined by a naming convention including node name;

v. feature: block; pattern mapping: a <scope> activity with two <empty> activities nested within the <scope> activity to represent the input and output nodes in the block;

vi. feature: iteration; pattern mapping: a <while> activity having an attribute condition equivalent to the loop condition in the loop node of the iteration; two <empty> activities nested within the <while> activity to represent input and output nodes in the loop body of the iteration;

vii. feature: receive event; pattern mapping: a <pick> activity containing <onMessage> structures to define events accepted by the <pick> activity, corresponding to events defined in the receive event;

viii. feature: compensation; pattern mapping: a <compensationHandler> structure comprising an activity within the structure to compensate an execution failure;

ix. feature: correlation; pattern mapping: a <correlation> element having a correlation ID defined and referenced by a <correlationSet> element; the <correlation> element being nested within a <receive> activity representing an input node, and within all <pick> activities corresponding to one or more receive event nodes;

x. feature: variables; pattern mapping: containers;

xi. feature: fault handling; pattern mapping: a <catch> structure containing elements in a fault path if the fault is only thrown once; where the fault is capable of being repeatedly caught and thrown then

(a) if thrown internally: a <throw> activity; or

(b) if thrown externally: a <reply> activity; and

xii. feature: transition condition; pattern mapping: an attribute in a <source> element of a <link> element representing the transition[[;]] .

23. (Currently Amended) A computer implemented method for converting from graphical to structural ~~textbased~~ text-based representations of business processes, the method comprising :

defining and maintaining a data representation of a set of features identifiable in graphical business process representations, each feature in the set of features having an associated pattern mapping defined relative to structural text-based representations,

identifying portions of an initial graphical representation matching features in the set of features, and

generating structural text-based representations of the identified portions of the initial graphical representation by applying the pattern mappings associated with the matching features to the identified portions of the graph-based representation, wherein the generated structural text-based representations are stored in a computer memory.

24. (Original) The method of claim 23, in which the set of identifiable features comprises features selected from: synchronous and asynchronous processes, request/response activities, one-way activities, empty nodes, blocks, iterations, receive events, compensation, correlation, variables, fault handling and transition conditions.

25. (Currently Amended) The method of claim 24, in which the set of identifiable features and the associated pattern mappings, comprises at least five feature and pattern mapping pairs selected from the .following set of pairs:

i. feature: synchronous/asynchronous processes; pattern mapping: a synchronous process representation comprises a <receive> activity as its input interface, and a <reply> activity as its output interface; an asynchronous process representation comprises a <receive> activity as its input interface, and an <invoke> activity as its output interface;

ii. feature: request/response activity; pattern mapping: an <invoke> activity with attributes inputContainer and outputContainer to specify input and output containers assigned to the activity;

iii. feature: one-way activity; pattern mapping: an <invoke> activity, with attribute inputContainer and no outputContainer;

iv. feature: empty node; pattern mapping: an <empty> activity defined by a naming convention including node name;

v. feature: block; pattern mapping: a <scope> activity with two <empty> activities nested within the <scope> activity to represent the input and output nodes in the block;

vi. feature: iteration; pattern mapping: a <while> activity having an attribute condition equivalent to the loop condition in the loop node of the iteration; two <empty> activities nested within the <while> activity to represent input and output nodes in the loop body of the iteration;

vii. feature: receive event; pattern mapping: a <pick> activity containing <onMessage> structures to define events accepted by the <pick> activity, corresponding to events defined in the receive event;

viii. feature: compensation; pattern mapping: a <compensationHandler> structure comprising an activity within the structure to compensate an execution failure;

ix. feature: correlation; pattern mapping: a <correlation> element having a correlation ID defined and referenced by a <correlationSet> element; the <correlation> element being nested within a <receive> activity representing an input node, and within all <pick> activities corresponding to one or more receive event nodes;

x. feature: variables; pattern mapping: containers;

xi. feature: fault handling; pattern mapping: a <catch> structure containing elements in a fault path if the fault is only thrown once; where the fault is capable of being repeatedly caught and thrown then

(a) if thrown internally: a <throw> activity; or

(b) if throw externally: a <reply activity; and

xii. feature: transition condition; pattern mapping: an attribute in a <source> element of a <link> element representing the transition[[;]] .

26. (Previously Presented) The method of claim 24, in which the generating of the structural text-based representations of the identified portions of the initial graph-based representation further comprises converting Java code referenced in the initial graphical representation to XPath code in the generated structural text-based representation.

27. (Previously Presented) The method of claim 26, in which the converting of the Java code to the XPath code comprises converting Java snippet nodes, and Java assignment and condition expressions.

28-34. (Cancelled)